International Journal of Computer Science Engineering
and Information Technology Research (IJCSEITR)
ISSN(P): 2249-6831; ISSN(E): 2249-7943
Vol. 5, Issue 1, Feb 2015, 75-90
© TJPRC Pvt. Ltd.

POLVM: PARALLEL OPTIMIZED LIVE VM MIGRATION

JAYASREE P[1] & MEENAKSHISHARMA[2]

[1]M.Tech, Department of CSE, Sri Sai College of Engineering and Technology, Badhani, Pathankot, Punjab, India

[1]Assistant Professsor, GEC Bartonhill, Trivandrum, Kerala, India

[2]Professor, Sri Sai College of Engineering and Technology, Badhani, Pathankot, Punjab, India

ABSTRACT

Live VM migration is a verydominant tool for cluster administrators, allowing parting of hardware and software considerations, consolidating clustered hardware into a single coherent management domain and facilitates fault management, load balancing. Live Migration is a planned scenario. In live migration move a running virtual machine from one host to another host with no perceived downtime for the VM. During migration there maintain TCP connections of the guest OS. VM is not aware of the migration and is considered as black box. Current system is based on future load prediction mechanism. VM resource allocation and optimized migration is done on this factor. During this VM migration, there is no suitable criteria for unique identification and location of VM that means which VM is migrated and where to be migrated. POLVM is based on a Cloud Booster Algorithm. In this system VM resource allocation mechanism carried out through by considering both Node weights and future prediction. To produce a better approach for solving the problem of VM resource migration in a cloud computing environment, this paper demonstrates Adaptive Genetic Algorithm based on VM resource migration strategy to accomplish system load balancing.

KEYWORDS: Cloud Computing, Optimized Migration, Adaptive Genetic Algorithm, Cloud Booster Algorithm, Virtual Machine

I. INTRODUCTION

The cloud is a next generation platform that provides dynamic resource pools, virtualization, and high availability. Today, we have the ability to utilize scalable, distributed computing environments within the confines of the Internet a practice known as cloud computing. Cloud computing is the concept implemented to decipher the daily computing problems, likes of hardware software and resource availability unhurried by computer users. The cloud computing platform guarantees subscribers that it sticks to the service level agreement (SLA) by providing resources as service and by needs. However, day by day subscriber needs are increasing for computing resources and their needs have dynamic heterogeneity and platform irrelevance. But in the cloud computing environment, resources are shared and if they are not properly distributed then it will result into resource wastage.

Another essential role of cloud computing platform is to dynamically balance the load amongst different servers in order to avoid hotspots and improve resource utilization. Therefore, the main problems to be solved are how to meet the needs of the subscribers and how to dynamically and efficiently manage the resources.

Live OS migration is an extremely powerful tool for cluster administrators, allowing separation of hardware and software considerations, and consolidating clustered hardware into a single coherent management domain. If a physical machine needs to be removed from service an administrator may migrate OS instances including the applications that they are running to alternative machine(s), freeing the original machine for maintenance. Similarly, OS instances may be rearranged across machines in a cluster to relieve load on congested hosts. In these situations the combination of virtualization and migration significantly improves manageability. In case of overloading, migration process can be performed on VMs running on single server and it can efficiently utilize the available resources. A much more efficient scheduling technique is being used which will allow for proper migration of VM processes holding good for green computation.

Resources are shared in the cloud computing environment and if they are not properly distributed then the result will be resource wastage. In Infrastructure-as-a-Service (IaaS) clouds, dynamic resource allocation is exploiting by the parallel data processing frame work [1]. This system fits well in a cloud for efficiently parallelizing incoming set of tasks using large data. [2] The key concept in enabling the "computing-as-a-service" vision of cloud- based solutions is Virtualization. Efficiency of resource usage and dynamic resource provisioning capabilities [3] of the VM are improved by the help of VM related features such as flexible resource provisioning and live migration.

A. Advancement in Migration Algorithms

The migrations can be classified into two namely: compulsory migration and split-warm migration. The compulsory migration is the one which is happening due to the overloaded servers (i. e, hot spot migration). The optimized resource utilized VM is the Warm VM. The maximum attempt will be to increase the warm range (the difference between the hot threshold and cold threshold). The split-warm migration is happening in between the warm VMs for assuring the optimized resource utilization. The distribution of the resources in the system is being done randomly amongst the VMs. So the resources can be over allocated or under allocated in the VMs. Thus POLVM proposed an efficient method so as to ensure the proper utilization of the resources during migration.

B. Requirement of Adaptive Genetic Algorithm

Adaptive Genetic algorithm is a random searching method that has a better optimization ability and internal implicit parallelism. It focuses on the system load balancing. If overloading occurs, then it helps VM migrate from overloaded physical machine to other, where the VM can run effectively. With the advantages of adaptive genetic algorithm, this paper presents effective scheduling strategy of VM resources in cloud computing environment.

II. LOAD BALANCING

A. Load Balance in IP Networks

These steps should be used to configure TCP/IP only if you choose not to use Network Load Balancing Manager to create your cluster. When configuring Network Load Balancing, you can use either Network Load Balancing Manager, or you can access the Network Load Balancing Properties dialog box through Network Connections. It is essential that you enter the dedicated IP address first, in the Internet Protocol (TCP/IP) Properties dialog box (not in the Advanced TCP/IP Settings dialog box), so that all outbound connections made from this host (for example, Telnet or FTP) are initiated with this address. TCP/IP is the only network protocol that should be present on the cluster adapter. You must not add any other protocols (for example, IPX) to this adapter. Be aware that if you are using Network Load Balancing to load balance Point-to-Point Tunneling Protocol (PPTP) servers, you must use two network adapters on each cluster host although Network Load Balancing does not need to be installed on both adapters. When load balancing PPTP, the hosts cannot have a dedicated IP addresses configured.

B. Load Balance in Cloud

The goals of cloud balancing in cloud are similar to those associated with traditional GSLB (Global Server Load Balancing) ensure the availability[10] of applications while simultaneously maximizing performance, regardless of the location or device from which users are accessing the application. Whether that access point is within an organization's data center utilizing private cloud resources or via a cloud provider, DNS requests are sent to the most appropriate location. These technical goals are met through a combination of application and network awareness and collaboration between the global application delivery solution and local load balancing solutions. By coordinating across application deployments in multiple data centers, whether in the cloud or traditionally based, organizations can, through careful monitoring of capacity and performance-related variables, achieve optimal application performance while ensuring availability.

C. Load Balance through Migration

Individual hosts have finite hardware resources, and are susceptible to failure. To mitigate against failure and resource exhaustion, hosts are grouped into clusters, which are essentially a grouping of shared resources. The Manager is able to ensure that no single host in a cluster is responsible for all of the virtual machines in that cluster. Conversely, the Manager is able to recognize an underutilized host, and migrate all virtual machines off of it, allowing an administrator to shut down that host to save power.

Available resources are checked as a result of three events:

- **Virtual Machine Start** - Resources are checked to determine on which host a virtual machine will start.
- **Virtual Machine Migration** - Resources are checked in order to determine an appropriate target host.
- **Time Elapses -** Resources are checked at a regular interval to determine whether individual host load is in compliance with cluster load balancing policy.

The Manager responds to changes in available resources by using the load balancing policy for a cluster to schedule the migration of virtual machines from one host in a cluster to another. Load balancing policy is set for a cluster, which includes one or more hosts that may each have different hardware parameters and available memory. The load balancing process runs once every minute for each cluster in a data center. It determines which hosts are over-utilized, which hosts under-utilized are, and which are valid targets for virtual machine migration. The determination is made based on the load balancing policy set by an administrator for a given cluster. There are three load balancing policies: None**,** Even Distribution and Power Saving**.**

- **Load Balancing Policies- None**: If no load balancing policy is selected, virtual machines are started on the host within a cluster with the lowest CPU utilization and available memory. To determine CPU utilization a combined metric is used that takes into account the virtual CPU count and the CPU usage percent.
- **Load Balancing Policies-Even Distribution**: An even distribution load balancing policy selects the host for a new virtual machine according to lowest CPU utilization. The maximum service level is the maximum CPU utilization that is allowed for hosts in a cluster, beyond which environment performance will degrade. The even distribution policy allows an administrator to set a maximum service level for running virtual machines. Host resources are checked once per minute, and one virtual machine is migrated at a time until the host CPU utilization is below the maximum service threshold.

- **Load Balancing Policies: Power Saving:** A power saving load balancing policy selects the host for a new virtual machine according to lowest CPU utilization. The maximum service level is the maximum CPU utilization that is allowed for hosts in a cluster, beyond which environment performance will degrade. The minimum service level is the minimum CPU utilization allowed before the continued operation of a host is considered an inefficient use of electricity. The even distribution policy allows an administrator to set a maximum and minimum service level for running virtual machines. If a host has reaches the maximum service level and stays there for more the set time, the virtual machines on that host are migrated one by one to the host that has the lowest CPU utilization. The process continues until the host CPU utilization is below maximum service level. If a host CPU utilization falls below the minimum service level the virtual machines are migrated to other hosts in the cluster if their maximum service level permits. When an under-utilized host is cleared of its remaining virtual machines, it can be shut down by an administrator to preserve power.

III. SYSTEM MODELAND OPTIMIZATION OBJECTIVES

In this section, the system model and the optimization objectives that are mean file unavailability, mean service time, load variance, energy consumption and latency are presented in detail.

A. System Model

The cloud storage cluster supports an efficient method for sharing data and other resources, and it focuses on providing and publishing storage service on the Internet which is sensitive to application workloads and user behaviors. The key component of the cloud storage cluster is the distributed file system. Google File System (GFS), Hadoop Distributed File System (HDFS) and Amazon Simple Storage Service (S3) are three famous examples.

Here, we suppose that the cloud storage cluster is composed of m independent heterogeneous data nodes PM_1,$PM_{2,}$ ... ,PM_j, ... ,PM_m storing a total of n different files f_1, ... ,f_i, ... ,f_n. Fig. 1 shows the system model of our parallel optimized Live Migration.

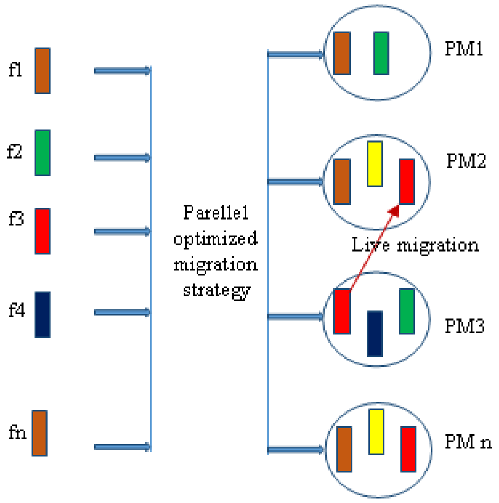


Figure 1: System Model

The optimized migration strategy is used to distribute n files into m data nodes. We assume that each access to file f_i is a sequential read of the entire file, which is a typical scenario in most file systems [1]. And we do not consider file partitioning, and thus, each file must be allocated entirely onto one data node. This does not restrict the generality of our scheme as each file partition can be treated as a stand-alone file. So the problem we addressed is statically or dynamically assigning no partitioned files in a cloud storage cluster where the file accesses exhibit Poisson arrival rates and fixed service times. In this article, we only consider that all the data are ‘‘write-once, read-many’’ [2], and thus no data consistency mechanism is needed.

B. Optimization Objectives

Multi-objective optimization problem [5] consists of optimizing a set of objectives. Some of which might be conflicting with one another. For example achieving better mean file unavailability might require creating additional migrations. Similarly, placing migrated replicas on the data servers with the lowest disk space utilization might not be ideal from the latency minimization perspective. The problem on hand deals with these conflicting objectives by evolving a set of solutions that compromise these conflicting objectives.

The quality of solutions obtained by parallel optimization is inherently dependent upon how well the objectives are formulated. In this section, we model each of the objectives in detail.

- **Mean File Unavailability (MFU)**

The first objective of the cloud storage cluster is to provide the highest availability for files. Let the decision variable U(i,j) equals to 1 if the file f $_i$exists on data node D_j, otherwise it equals to 0. Let us denote p $_j$ as the failure probability of data node $D_j(1 \leq j \leq m)$. Failure probabilities of data nodes are drawn randomly and they can be updated by tracking the history of data nodes in the system. As each file has some replicas and they are distributed in different data nodes. The whole file will be unavailable if and only if all replicas of this file are unavailable, and all replicas are independent of each other. So the probability of file f_i unavailable is defined as follows. So the probability of file f_i unavailable is defined as follows:

$$P\left(\bar{fi}\right) = \prod_{j-1}^{m} \emptyset(i,j) \times Pj \tag{1}$$

The availability of file f_i can be calculated by

$$P(fi) = 1 - \prod_{j-1}^{m} \emptyset(i,j) \times Pj \tag{2}$$

Mean file unavailability objective function MFU is modeled through the following function:

$$\mathrm{MFU}=\frac{1}{n} \times \sum_{i-1}^{n} P\left(\bar{fi}\right) \tag{3}$$

- **Mean Service Time (MST)**

The secondary objective is to minimize mean service time which describes the ability of the system process rate. Less mean service time means faster process ability of the storage system. Placing popular files in data nodes with better performance and infrequently accessed files in those data nodes with relatively weak performance is able to improve the average service time of the system. Let st(i,j) be the expected service time of file f_i on the data node D_j

$$\bar{St}\,(\mathrm{i,j})=\emptyset(i,j) \times \frac{si}{tpj} \tag{4}$$

Where Si is the size of file fi and TPj is the transfer rate of the data node Dj

$$MST=\frac{1}{n}\times\sum_{i-1}^{n}(\bar{St})\,(i) \qquad (5)$$

- **Load Variance (LV)**

Data node load variance is the standard deviation of data node load of all data nodes in the cloud storage cluster which can be used to represent the degree of load balancing of the system. The lower value of load variance, the better load balancing is achieved. Since the combination of access rate and service time of the file f_i accurately gives the load of it, the load l(i,j) of f_i which is on the data node $D_j$is defined as follows :

The load of data node Dj can be calculated by

$$l(j)=\sum_{i-1}^{n} l(i,j) \qquad (6)$$

So the mean load of the system can be expressed by

$$L=\frac{1}{m}\sum_{j-1}^{n} l(j) \qquad (7)$$

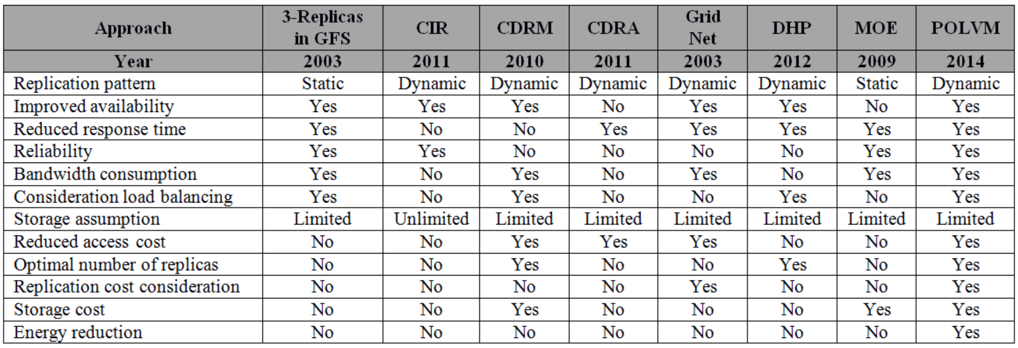
- **Energy Consumption (EC)**

The total energy consumption[8] is mainly composed of renewable energy consumption (RE) and cooling energy consumption (CE). RE and CE are to be minimized. Recent studies has shown that the power consumption by servers can be accurately described by a linear relationship between the power consumption and utilization. In this article, we utilize real data on power consumption provided by the results of the SPEC power benchmark.

Let Q denote the coefficient of performance (COP) in data node D_j. Under the principles of thermodynamics, COP highly depends on the indoor and outdoor temperature (T_{in} and T_{out}). Q is defined as follows:

$$Q=\frac{1}{\frac{Tout}{Tin}-1} \qquad (8)$$

Table 1: Features Studied in the Related Work POLVM



Approach	3-Replicas in GFS	CIR	CDRM	CDRA	Grid Net	DHP	MOE	POLVM
Year	**2003**	**2011**	**2010**	**2011**	**2003**	**2012**	**2009**	**2014**
Replication pattern	Static	Dynamic	Dynamic	Dynamic	Dynamic	Dynamic	Static	Dynamic
Improved availability	Yes	Yes	Yes	No	Yes	Yes	No	Yes
Reduced response time	Yes	No	No	Yes	Yes	Yes	Yes	Yes
Reliability	Yes	Yes	No	No	No	No	Yes	Yes
Bandwidth consumption	Yes	No	Yes	No	Yes	No	Yes	Yes
Consideration load balancing	Yes	No	Yes	No	No	Yes	No	Yes
Storage assumption	Limited	Unlimited	Limited	Limited	Limited	Limited	Limited	Limited
Reduced access cost	No	No	Yes	Yes	Yes	No	No	Yes
Optimal number of replicas	No	No	Yes	No	No	Yes	No	Yes
Replication cost consideration	No	No	No	No	Yes	No	No	Yes
Storage cost	No	No	Yes	No	No	No	Yes	Yes
Energy reduction	No	No	No	No	No	No	No	Yes

Therefore, energy consumption objective function EC is given by the following

$$EC = 1+\frac{1}{Q} \qquad (9)$$

- **Mean Latency (ML)**

Minimizing latency is important for any storage system. Minimizing latency depends on utilizing high bandwidth channels, as high bandwidth channels yield lesser latency. Thus, it is better to place popular files in data nodes with high bandwidth to minimizing their latency.

$$ML= \sum_{i-1}^{n} Li/n \quad (10)$$

PROPOSED SYSTEM

A. Parallel Optimized Live VM Migration

Our scheme depends on taking the historical system information and feed it to an Virtual Machine where we try not only to keep file availability, mean service time, load variance, energy consumption and latency within constraints, but in addition we try to optimize file availability, mean service time, load variance, energy consumption and latency in order to find different trades offs between these objectives. Since there is more than one objective, we are doing Adaptive genetic optimization to find the optimal Load which has to be migrated and achieve efficient job allocation and Load Balance. Recent studies has shown that the power consumption by servers can be accurately described by a linear relationship between the power consumption and utilization.

B. System Architecture

Proposed system architecture design create and implement Cloud booster algorithm for VM allocation which reduce hot spot nodes in the cloud and to design and implement the managing of overloading process to Migrate VM from overloaded PM to overcome overloading. The system design is shown in Fig.2. In Client Node the user can submit task to the Cloud Controller, which distribute the job among parallel nodes. Job Allocator is responsible for allocating the job among the parallel nodes. While allocation it will consider the results from the future predictor. After the allocation of jobs to the cloud nodes. They will do the jobs independently in cloud nodes. The results will be sent back to the cloud controller. A future predictor is need to predict the future resource needs of VMs. One solution is to look inside a VM for application level statistics by parsing logs of pending requests. Doing so requires modification of the VM which may not always be possible. Instead of making our prediction based on the past external behaviors of VMs. It measures the load on every minute and predicts the load in the next minute. Based on the prediction result, the given jobs will be distributed to the cloud nodes.

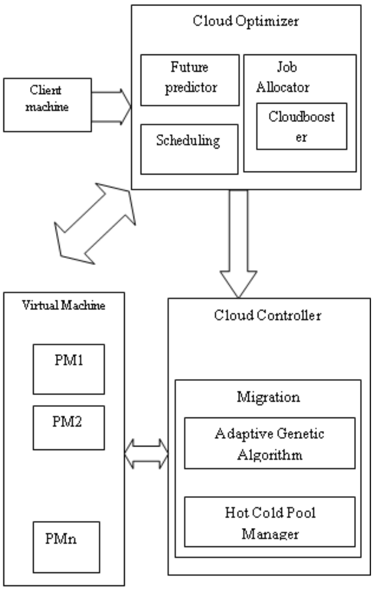


Figure 2: Proposed System Architecture Design

- Cloud Node (Requestor) gets the job request from user to cloud controller.
- Cloud booster algorithm will calculate current allocation & Node capabilities.
- Future predictor will predict the future loads based on the allocation details from the job allocator.
- Job allocator allocates the jobs based on the allocation sequence from the cloud booster.
- Hot cold pool manager gets the allocation details from the job allocator, and finds which all nodes are in hot and cold threshold.
- Based on the node load details from the Hot cold pool manager, the Migration manager overload detector will finds the victim job and remote node, where it can be able to migrate the victim job.
- PMs gives the results back to user through cloud controller.

C. Scheduling

Scheduling [6, 7] run through cost model. Let consider the following cost factor: $\omega_i$be the cost per instruction for processing unit i and β_j indicates the delay cost of job j. Suppose, M machines with N jobs and assign these N jobs into M machines (N=M), in such an order that following condition can be satisfied from user side, finish time ($T f$) must be less than the worst case completion time ($Twcc$), scheduling must be done such way to preserve QoS and to avoid possible starvation as:

$$T f \leq Twcc \tag{11}$$

This condition must be satisfied anyhow, otherwise the job is considered as a failure job and the corresponding scheduling is illegal. From cloud provider side, to minimize the cost spend on the job: Suppose $i^{th\ s}$ machine is assigned to j^{th} job. Then the cost for execution job j is:

$$IC_j * \omega_i \tag{12}$$

Where IC is the instruction count. Let Ψj, estimated delay cost for job j, can be defined as:

$$\begin{cases} 0 \quad if \quad Td \geq T f \\ \Psi j = \delta j * (T f - Td) \quad if \quad Td < T f \end{cases} \tag{13}$$

where, Td is the deadline for job j and Tf is the estimated finish time, when job j is assigned to processing unit i. Thus overall cost to execute all M jobs can be given by:

$$C=\sum_{i=1}^{n}(ICj * \omega i) + \Psi j) \tag{14}$$

Thus, cloud provider's aim is to schedule jobs (i.e find a permutation: N→M such a way which minimize the value of:

$$Min(C) = \min [\sum_{i=1}^{n}(ICj * \omega i) + \Psi j)] \tag{15}$$

As there are M number of machines and N number of jobs and assuming that all machines are capable to perform any job, then there are total $M * N$ numbers of way to assignment. And if $M = N$, then it need M! assignments, which has an exponential complexity O (M!).

With the principle of scheduling all participating processors stopped computing at the same time instant, the closed-form formulas for both processing time and workload fraction for each processor are derived. We also consider cost-based multi- scheduling on a compute cloud environment.

In load scheduling domain [6] an optimality criterion [2] is used to derive an optimal solution as follows. It states that in order to obtain an optimal processing time, it is necessary and sufficient that all the sites that participate in the computation must stop at the same time. Otherwise, load could be redistributed to improve the processing time.

V. EFFICIENT JOB ALLOCATION AND MIGRATION

The objective of this paper is to design and implement Adaptive Genetic Algorithm [4] which manages the overloading process to Migrate VM from overloaded PM to overcome overloading and reduce hot spot nodes in the cloud and the Cloud Booster Algorithm provides a scheduling strategy to enable effective load balancing. When the loads are distributed to the nodes without considering their processing capacities it would affect the response time of the system.

A. Cloud Booster Algorithm

This algorithm has 3 phases.

- Node weight calculation phase.
- Load distribution or Job allocation phase.
- Execution of job in parallel processing framework.

In node weight calculation phase, Resource information of a particular machine is retrieved. Resource factors for calculating weight for a particular machine is tabulated as follows:

Table 2: Resource Information

Resource	Sample Value for a Machine	Weight Constant
CPU Speed(GHZ)	2.4	0.5
Main Memory(GB)	2	0.4
CPU Utilization	0.013	0.5
Memory Utilization	0.536	0.4

Here CPU Utilization and Memory usage are negative factors and the Available CPU and Memory are the positive factors. CPU has greater impact on the performance of a machine comparing to memory or other factors. So it has maximum weight constant. Weight for the above machine configuration is calculated as follows:

Memory Utilization = (Total Memory – Used Memory) / Total Memory

CPU Utilization = Total CPU Usage of Processes / No. of Processes

*Available CPU = Clock Speed * (1 – CPU Utilization)*

*Available Memory = Main memory * (1 – Memory Utilization)*

For converting each parameter into same scale between 0 and 1, divide each value with the maximum value. Then weight for each resource is separately determined.

*CPU Weight (WCPU) = Available CPU * Weight constant for CPU*

*Memory Weight (WMem) = Available Memory * Weight constant for Memory.*

*Weight for current load (WLoad) = Current load * Weight constant for load*

Weight of Machine = WCPU + WMem – WLoad

In Load distribution phase [9] select the best parallel machine for job allocation. Here job Size is considered as load. Before job allocation, find the job percentage required by each machine. This is calculated by using the formula.

Required job percentage for a machine say X, $px = Wx / \sum Wi$, here $\sum$ Wi *is total weight of all the nodes.*

After estimating Job volume for each machine based on their node weightage parallel execution of task is started in parallel processing system and output is generated.

B. Adaptive Genetic Algorithm

Algorithm 1: Adaptive Genetic Algorithm.

```
Initialize the parameters: Node weight, job allocation
Generate a population_sequence P randomly;
generation ← 1;
while generation ≤ max_gendo
    Clear the new population_sequence P';
    Use a Load Predictor function f(·) to evaluate each individual Weight in P;
while|P'| ≤ N do
        Select four sequence from P;
        Perform crossover with rate pc;
        Perform mutation with rate pm;
        Insert the offspring_sequence to P';
endwhile
        P ← P';
        generation ← generation + 1;
endwhile
```

Initial Sequence - Here we generate n chromosomes randomly. The following three steps can be iterated for n times. Consider 4 sequences and do the following steps.

- **Selection**
  - Generate 2n sequences.

- First copy n sequences from initial sequences.
- Generate n remaining sequences as randomly select sequences from initial sequences with best fitness value. Repeat these n times then we get 2n sequences.

Generate 8 Sequences by copying above 4 and Generate other 4 as follows. Generate two random number r1, r2 if rank of r1 < r2 select r1 otherwise select r2. For example r1=4 and r2=3 then select r2 because rank of r2 < r1 and set as sequence 5.

- **Cross Over**
  - Generate 2n sequences.
  - First copy n sequences from selection sequences.
  - Remaining n sequences are generated as combining the “selection sequences”
- **Mutation**
  - Generate 2n sequences.
  - First copy n sequences from Cross over sequences.
  - Remaining n sequences is generated as randomly select a sequences from above & Perform some changes in the sequence.

Sort these sequences on rank base, select first n sequences with min weight. Sort these 8 sequences on rank base, select first 4 sequences with minimum weight.

$$R = \sum_{i=1}^{N_{vm}} \frac{\frac{Wi}{Max(Wi)}}{N_{vm}} + \sum_{i=1}^{N_{vm}} \frac{\frac{Max(Pi)-Pi}{Max(Pi)}}{N_{vm}} + \sum_{i=1}^{N_{vm}} \frac{\frac{Ti}{Max(Ti)}}{N_{vm}} + \sum_{i=1}^{N_{vm}} \frac{\frac{Min(Ci)-Ci}{Max(Ci)}}{N_{vm}} \tag{16}$$

R - Rank

N_{vm} - Number of jobs

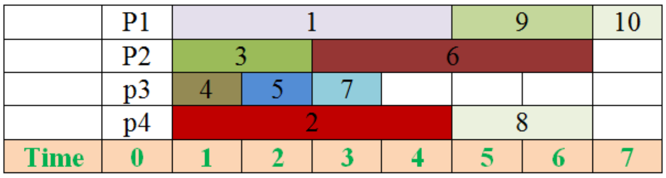
W_i - Weighting time of ith Job

P_i - Communication Cost of ith Job

T_i - Idle time ith VM

C_i - Migration cost of ith Job

Consider Following Allocation



	P1	1				9		10
	P2	3		6				
	p3	4	5	7				
	p4	2				8		
Time	0	1	2	3	4	5	6	7

Figure 3: Job Allocation

Waiting Time of Job 9: 6

Here highest waiting time 7 (job 10)

*PW (Percentage of waiting time) = Sum (waiting time for each job) / (highest waiting time * number of jobs)*

Idle time of p4=1, Highest idle time=4 (p3)

*PI (Percentage of idle time) = Sum (idle time for each node) / (highest idle * number of nodes)*

Suppose p3 is considered as costly node with 10 Rs/sec, But Job j2 is allocated to p4 (have 5 Rs/Sec) and job length=4 sec, Income=4*5=20. Suppose this job run in p3, income=4*10=40

So profit lose=40-20=20.

*PC(percentage of communication cost)=Sum (communication cost of each node) / (Maximum communication cost * number of jobs).*

*PP(percentage of replication cost)=Sum(replication cost of each node) / (Maximum replication cost * number of jobs).*

Rank= PC + PP + PW + PI

Here, we select sequence with minimum rank. In this way Adaptive Genetic Algorithm generates optimized new sequence. During this generation, AGA minimizes rank. Here rank is the sum of waiting time, communication cost, replication cost and idle time. So these generations minimize above factors.

VI. IMPLEMENTATION

In this section, we evaluate and analyze the performance of different Virtual Machine by the use of cloudsim.

A. The Expression of Load

The summation of all the running VMs on a physical machine can be termed as the load of the physical machine. If we assume that the T is the best time duration to monitor historical data i.e., from current time to the last T minutes will be the historical data zone, which will be used to solve the load balancing problem.

By using the variation law of the physical machine load, we can split T into n subsequent time intervals. Therefore, T becomes [(t1- t0), (t2 - t1),…,(tn - tn-1)].

The time interval k can be defined as (tk – tk-1). If the load of a VM is stable in all the periods then V (i, k) refers to the load of VM i in the interval k.

By using the above definition, we can define the average load of VM on physical server Pi in time cycle T is

$$Vi(i,T) = \frac{1}{T}\sum_{k=1}^{n} V(i,k)x\left(t_k - t_{k-1}\right) \quad (17)$$

By using the above average load definition of VM, we can calculate the load of the physical machine for last T interval by adding all the loads of the VMs present on the physical machine. So, the expression to compute the load of physical machine Pi is as follows:

$$P(i,T) = \sum_{j=1}^{mi} V_i(j,T) \quad (18)$$

The virtual machine V needs to be deployed on the current system. We have the resource information [8] for Virtual Machine V, and from that we can estimate the load of the VM as V’. Therefore, when the VM V is joined to every physical server, we can calculate the load of each Pi as follows:

$$P(i,T)' = \begin{cases} P(i,T) + V', AfterDeployV \\ \quad P(i,T), \wedge\, Others \end{cases} \tag{19}$$

B. RESULTS

The Cloud Booster algorithm provides an effective dynamic load balancing method in heterogeneous cloud environments. In order to test the performance, a graph is plotted with Job size or load on X axis and response time on Y axis.

Parallel processing setup is constructed with 3 different machines. Each machines having different processing capacity. Data sets are obtained from giving inputs of different job size and response time is noted. From the above data, performance can is plotted separately for Cloud booster and it is clear that as load increases, proposed system performed exceptionally well and response time of job is considerably reduced. In existing system, as load increases response time is also increased linearly. That is response time is very high for task having larger job size. To analyse the behaviour of the adaptive genetic algorithm with VM scheduling, we got a scheduling details within 0.5 micro seconds for following inputs. After a no of rounds of various inputs, we can realize that as number of rounds increases we got better scheduling data, i.e. scheduling job with minimum waiting time, Consumer cost, provider idle time and maximum provider benefit. The VM loads are balanced on every physical node and also the system load variation parameter is also achieved. As a result, we can conclude that the adaptive genetic algorithm has quite a better globally astringency and it can come closer to be the best solution in very little time. These results are shown in Fig. 6.

We can observe that by using a dynamic ratio load balancing favoring the faster provider, the overall network throughput is higher. This ensures that all of the download and upload operations complete within a nearly similar time frame, and hence the “tail time” is effectively mitigated. As a result, the wireless radio of the tested mobile device can switch to idle state earlier, which could lead to lower battery energy consumption.

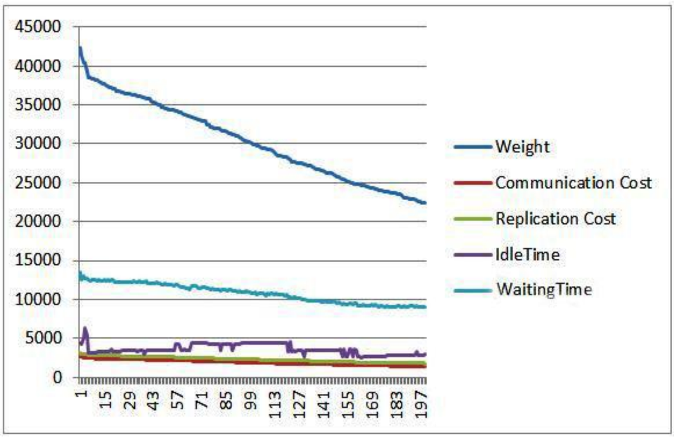


Figure 4: Performance Diagram for Different Parameters Against Number of Iteration

VII. CONCLUSIONS

Proposed system analyze the parallel optimization problem for Live VM migration in cloud. We introduce an

Adaptive Genetic Algorithm to characterize the service process in which VM should migrate to where in cloud and achieve load balance. The proposed system overloaded nodes has to be optimized and the resources in under loaded node can be redistributed amongst the other resource-starving nodes. After performing proper migration, the underutilized node can be turned off and this will in turn contribute towards more energy conservation. It allows business customers to scale up and down their resource usage based on needs. Different organization provides same service with different service charges and waiting time. So customers can select services from these cloud providers according to their criteria like cost and waiting time. We have presented the design, implementation, and evaluation of Parallel Optimized Live Migration for an efficient resource management in cloud computing. We use Cloud Booster Algorithm for finding the node's capabilities and job allocation and Adaptive Genetic Algorithm for VM migration. Our algorithm achieves both overload avoidance and energy saving for systems with multi-resource constraints. In future enhancement will propose a new algorithm for resource scheduling and comparative with existing algorithms for reducing the number of migration.

REFERENCES

1. Zhen Xiao, Senior Member, IEEE, Weijia Song, and Qi Chen "Dynamic Resource Allocation Using Virtual Machines for Cloud Computing Environment" IEEE Transaction on Parallel and Distributed Systems Year 2013
2. A Ajitha, D Ramesh "Improved Task Graph-Based Parallel Data Processing for Dynamic Resource Allocation in Cloud"International Conference On Modelling Optimization And Computing ELSEVIER Volume 38, 2012, Pages 2172–2178.
3. Mayank Mishra, Anwesha Das, Purushottam Kulkarni, and AnirudhaSaho"Dynamic Resource Management Using Virtual Machine Migrations", Cloud Computing: Networking and Communication Challenges IEEE- 2012.
4. Michael Maurer, IvonaBrandicRizos "AdaptiveResourse Configuration for cloud Infrastructure Management" Future Generation Computer SystemsELSEVIER Volume 29, Issue 2, February 2013, Pages 472–487
5. Sai-Qin Long, Yue-Long Zhao, Wei Chen, Senior Member, IEEE, Weijia Song, and Qi Chen "MORM: A Multi-objective Optimized Replication Management strategy for cloud storage cluster" Journal of Systems Architecture ELSEVIER 12 December 2013(234–244).
6. Ghribi C, Hadji M and D Zeghlache, "Energy Efficient VM Scheduling for Cloud Data Centers: Exact allocation and migration algorithms"Cluster, Cloud and Grid Computing (CCGrid), 2013 13th IEEE/ACM International Symposium on May 2013
7. Monir Abdullah, Muhamwd Othman "Cost Based Multi-QoS job Scheduling using Divisible Load Theory in Cloud Computing" International Conference on Computational Science, ICCS 2013.
8. Dapeng Dong and John Herbert, "Energy Efficient VM Placement Supported by Data Analytic Service", IEEE/ACM International Symposium-May2013
9. MayankaKatyal, and Atul Mishra "Application of Selective Algorithm for Effective Resource Provisioning in Cloud Computing Environment" International Journal on Cloud Computing: Services and Architecture (IJCCSA),Vol. 4, No. 1, February 2014
10. Shabnam Khan "A Survey on Scheduling Based Resource Allocation in Cloud Computing" International Journal

for Technological Research in Engineering Vol. 1, Issue. 1, Sep – 2013